SVRE: NEW METHOD FOR TRAINING GANs

GAUTHIER GIDEL

Mila, Université de Montréal
Research intern at Element AI

GENERATIVE MODELING AND MODEL-BASED REASONING FOR ROBOTICS AND AI WORKSHOP

June 14, 2019

Reducing Noise in GAN Training with Variance Reduced Extragradient

Tatjana Chavdarova *

Gauthier Gidel *

François Fleuret

Simon Lacoste-Julien

* Equal contribution

Generative Adversarial Networks

[Goodfellow et al., 2014]

Challenges

- Standard supervised learning:

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$$

- GANs: ~~Hard~~ (different) optimization problem: minimax.

CHALLENGES

- Standard supervised learning:

$$\min_{\theta} \mathcal{L}(\theta)$$

- GANs: ~~Hard~~ (different) optimization problem: minimax.

$$\min_{\theta_G} \max_{\theta_D} V(\theta_G, \theta_D)$$

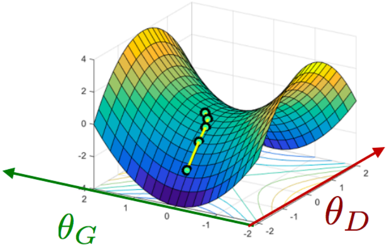
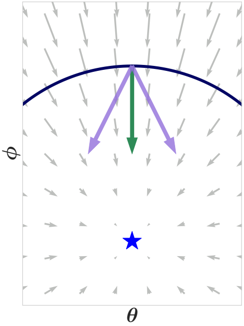


Image source: *Vaishnavh Nagarajan*

“Noise”: noisy gradient estimates
Due to stochasticity

- Using sub-samples (mini-batches) of the full dataset to update the parameters
- Variance Reduced (VR) Gradient: optimization methods that reduce such noise

Minimization: Single-objective

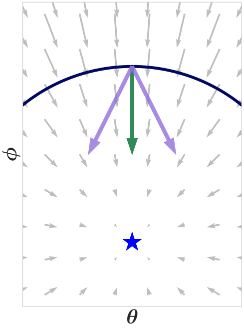


Batch method direction

Stochastic method direction: noisy

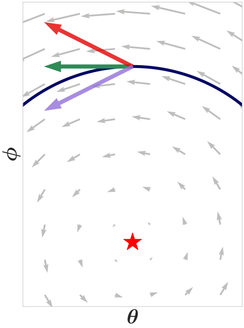
Variance Reduction–Motivation for Games

- Intuitively: **Minimization** *vs.* **Game** (Noise from Stochastic gradient)
- Empirically: **BigGAN**–"Increased batch size significantly improves performances"
- To sum up, two issues:



Minimization

"approximately" the right direction



Game

Direction with noise can be "bad"

Variance Reduction–Motivation for Games

- Intuitively: **Minimization** *vs.* **Game** (Noise from Stochastic gradient)
- Empirically: **BigGAN**–"Increased batch size significantly improves performances"
- To sum up, two issues:

Brock et al. [2018] report a relative improvement of 46% of the Inception Score metric [Salimans et al., 2016] on **ImageNet** if the mini-batch size is increased 8–fold.

Variance Reduction–Motivation for Games

- Intuitively: **Minimization** *vs.* **Game** (Noise from Stochastic gradient)
- Empirically: **BigGAN**–"Increased batch size significantly improves performances"
- To sum up, two issues:

  - Adversarial aspect from min-max $\rightarrow$ Extragradient.
  - Noise from stochastic gradient $\rightarrow$ Variance Reduction.

Extragradient

Extragradient

Two players $\boldsymbol{\theta}$, $\boldsymbol{\varphi}$. Idea: perform a "Lookahead step"

$$\text{Extrapolation:} \begin{cases} \boldsymbol{\theta}_{t+1/2} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_G(\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t) \\ \boldsymbol{\varphi}_{t+1/2} = \boldsymbol{\varphi}_t - \eta \nabla_{\boldsymbol{\varphi}} \mathcal{L}_D(\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t) \end{cases}$$

$$\text{Update:} \begin{cases} \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_G(\boldsymbol{\theta}_{t+1/2}, \boldsymbol{\varphi}_{t+1/2}) \\ \boldsymbol{\varphi}_{t+1} = \boldsymbol{\varphi}_t - \eta \nabla_{\boldsymbol{\varphi}} \mathcal{L}_D(\boldsymbol{\theta}_{t+1/2}, \boldsymbol{\varphi}_{t+1/2}) \end{cases}$$

Variance Reduced Gradient Methods

Variance Reduced Estimate of the Gradient

Based on Finite sum assumption:

$$\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(\boldsymbol{x}_i,\boldsymbol{\omega}),$$

Epoch based algorithm:

- Save the full gradient $\frac{1}{n}\sum_i \nabla\mathcal{L}(\boldsymbol{x}_i,\boldsymbol{\omega}^{\mathcal{S}})$ and the snapshot $\boldsymbol{\omega}^{\mathcal{S}}$.
- For one epoch use the update rule:

$$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} - \eta\Big[\underbrace{\nabla\mathcal{L}(\boldsymbol{x}_i,\boldsymbol{\omega})}_{\text{Stochastic gradient}} + \underbrace{\frac{1}{n}\sum_i \nabla\mathcal{L}(\boldsymbol{x}_i,\boldsymbol{\omega}^{\mathcal{S}}) - \nabla\mathcal{L}\left(\boldsymbol{x}_i,\boldsymbol{\omega}^{\mathcal{S}}\right)}_{\text{correction using saved past iterate}}\Big]$$

- Requires 2 stochastic gradients (at the current point and at the snapshot).
- If $\boldsymbol{\omega}^{\mathcal{S}}$ is close to $\boldsymbol{\omega}$ $\rightarrow$ close to full batch gradient $\rightarrow$ small variance.
- Full batch gradient expensive but tractable, *e.g.*, compute it once per pass.

Variance Reduced Estimate of the Gradient

Based on Finite sum assumption:

$$\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(\boldsymbol{x}_i,\boldsymbol{\omega}),$$

Epoch based algorithm:

- Save the full gradient $\frac{1}{n}\sum_i \nabla\mathcal{L}(\boldsymbol{x}_i,\boldsymbol{\omega}^{\mathcal{S}})$ and the snapshot $\boldsymbol{\omega}^{\mathcal{S}}$.
- For one epoch use the update rule:

$$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} - \eta\Big[\underbrace{\nabla\mathcal{L}(\boldsymbol{x}_i,\boldsymbol{\omega})}_{\text{Stochastic gradient}} + \underbrace{\frac{1}{n}\sum_i \nabla\mathcal{L}(\boldsymbol{x}_i,\boldsymbol{\omega}^{\mathcal{S}}) - \nabla\mathcal{L}\left(\boldsymbol{x}_i,\boldsymbol{\omega}^{\mathcal{S}}\right)}_{\text{correction using saved past iterate}}\Big]$$

- Requires 2 stochastic gradients (at the current point and at the snapshot).
- If $\boldsymbol{\omega}^{\mathcal{S}}$ is close to $\boldsymbol{\omega}$ $\rightarrow$ close to full batch gradient $\rightarrow$ small variance.
- Full batch gradient expensive but tractable, *e.g.*, compute it once per pass.

Variance Reduced Estimate of the Gradient

Based on Finite sum assumption:

$$\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(\boldsymbol{x}_i,\omega),$$

Epoch based algorithm:

- Save the full gradient $\frac{1}{n}\sum_i \nabla\mathcal{L}(\boldsymbol{x}_i,\boldsymbol{\omega}^{\mathcal{S}})$ and the snapshot $\boldsymbol{\omega}^{\mathcal{S}}$.
- For one epoch use the update rule:

$$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} - \eta\Big[\underbrace{\nabla\mathcal{L}(\boldsymbol{x}_i,\boldsymbol{\omega})}_{\text{Stochastic gradient}} + \underbrace{\frac{1}{n}\sum_i \nabla\mathcal{L}(\boldsymbol{x}_i,\boldsymbol{\omega}^{\mathcal{S}}) - \nabla\mathcal{L}\left(\boldsymbol{x}_i,\boldsymbol{\omega}^{\mathcal{S}}\right)}_{\text{correction using saved past iterate}}\Big]$$

- Requires 2 stochastic gradients (at the current point and at the snapshot).
- If $\boldsymbol{\omega}^{\mathcal{S}}$ is close to $\boldsymbol{\omega}$ $\rightarrow$ close to full batch gradient $\rightarrow$ small variance.
- Full batch gradient expensive but tractable, *e.g.*, compute it once per pass.

Variance Reduced Estimate of the Gradient

Based on Finite sum assumption:

$$\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(\boldsymbol{x}_i,\boldsymbol{\omega}),$$

Epoch based algorithm:

- Save the full gradient $\frac{1}{n}\sum_i \nabla\mathcal{L}(\boldsymbol{x}_i,\boldsymbol{\omega}^{\mathcal{S}})$ and the snapshot $\boldsymbol{\omega}^{\mathcal{S}}$.
- For one epoch use the update rule:

$$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} - \eta\Big[\underbrace{\nabla\mathcal{L}(\boldsymbol{x}_i,\boldsymbol{\omega})}_{\text{Stochastic gradient}} + \underbrace{\frac{1}{n}\sum_i \nabla\mathcal{L}(\boldsymbol{x}_i,\boldsymbol{\omega}^{\mathcal{S}}) - \nabla\mathcal{L}\left(\boldsymbol{x}_i,\boldsymbol{\omega}^{\mathcal{S}}\right)}_{\text{correction using saved past iterate}}\Big]$$

- Requires 2 stochastic gradients (at the current point and at the snapshot).
- If $\boldsymbol{\omega}^{\mathcal{S}}$ is close to $\boldsymbol{\omega}$ $\rightarrow$ close to full batch gradient $\rightarrow$ small variance.
- Full batch gradient expensive but tractable, *e.g.*, compute it once per pass.

SVRE: Variance reduction + Extragradient

Pseudo–algorithm

1. Save snapshot $\boldsymbol{\omega}^{\mathcal{S}} \leftarrow \boldsymbol{\omega}_t$ and compute $\frac{1}{n}\sum_i \nabla\mathcal{L}(\boldsymbol{x}_i, \boldsymbol{\omega}^{\mathcal{S}})$.
2. For i in $1, \ldots, \texttt{epoch\_length}$:
   - Compute $\omega_{t+\frac{1}{2}}$ with variance reduced gradients at ω_t.
   - Compute ω_{t+1} with variance reduced gradients at $\omega_{t+\frac{1}{2}}$.
   - $t \leftarrow t+1$
3. Repeat until convergence.

SVRE: Variance reduction + Extragradient

Pseudo–algorithm

1. Save snapshot $\omega^{\mathcal{S}} \leftarrow \omega_t$ and compute $\frac{1}{n}\sum_i \nabla \mathcal{L}(x_i, \omega^{\mathcal{S}})$.
2. For i in $1, \ldots,$ `epoch_length`:
   - Compute $\omega_{t+\frac{1}{2}}$ with variance reduced gradients at ω_t.
   - Compute ω_{t+1} with variance reduced gradients at $\omega_{t+\frac{1}{2}}$.
   - $t \leftarrow t + 1$
3. Repeat until convergence.

SVRE: Variance reduction + Extragradient

Pseudo–algorithm

1. Save snapshot $\omega^{\mathcal{S}} \leftarrow \omega_t$ and compute $\frac{1}{n}\sum_i \nabla \mathcal{L}(\boldsymbol{x}_i, \omega^{\mathcal{S}})$.
2. For i in $1, \ldots,$ `epoch_length`:
   - Compute $\boldsymbol{\omega}_{t+\frac{1}{2}}$ with variance reduced gradients at $\boldsymbol{\omega}_t$.
   - Compute $\boldsymbol{\omega}_{t+1}$ with variance reduced gradients at $\boldsymbol{\omega}_{t+\frac{1}{2}}$.
   - $t \leftarrow t+1$
3. Repeat until convergence.

SVRE: Variance reduction + Extragradient
Pseudo–algorithm

1. Save snapshot $\omega^{\mathcal{S}} \leftarrow \omega_t$ and compute $\frac{1}{n}\sum_i \nabla\mathcal{L}(\boldsymbol{x}_i, \omega^{\mathcal{S}})$.
2. For i in $1, \ldots,$ `epoch_length`:
   - Compute $\boldsymbol{\omega}_{t+\frac{1}{2}}$ with variance reduced gradients at $\boldsymbol{\omega}_t$.
   - Compute $\boldsymbol{\omega}_{t+1}$ with variance reduced gradients at $\boldsymbol{\omega}_{t+\frac{1}{2}}$.
   - $t \leftarrow t+1$
3. Repeat until convergence.

SVRE: Variance reduction + Extragradient

Pseudo–algorithm

1. Save snapshot $\omega^{\mathcal{S}} \leftarrow \omega_t$ and compute $\frac{1}{n}\sum_i \nabla \mathcal{L}(x_i, \omega^{\mathcal{S}})$.
2. For i in $1, \ldots,$ `epoch_length`:
    - Compute $\boldsymbol{\omega}_{t+\frac{1}{2}}$ with variance reduced gradients at $\boldsymbol{\omega}_t$.
    - Compute $\boldsymbol{\omega}_{t+1}$ with variance reduced gradients at $\boldsymbol{\omega}_{t+\frac{1}{2}}$.
    - $t \leftarrow t + 1$
3. Repeat until convergence.

SVRE: Variance reduction + Extragradient

Pseudo–algorithm

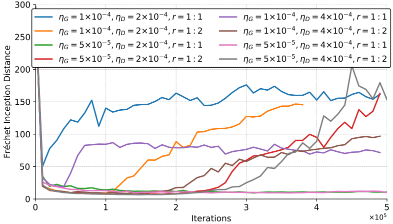
1. Save snapshot $\omega^{\mathcal{S}} \leftarrow \omega_t$ and compute $\frac{1}{n}\sum_i \nabla \mathcal{L}(x_i, \omega^{\mathcal{S}})$.
2. For i in $1, \dots,$ `epoch_length`:
   - Compute $\omega_{t+\frac{1}{2}}$ with variance reduced gradients at ω_t.
   - Compute ω_{t+1} with variance reduced gradients at $\omega_{t+\frac{1}{2}}$.
   - $t \leftarrow t + 1$
3. Repeat until convergence.

SVRE yields the fastest convergence rate for strongly convex stochastic game optimization in the literature.

SVRE: EXPERIMENTS

EXPERIMENTS
SVRE YIELDS STABLE GAN OPTIMIZATION

Stochastic baseline



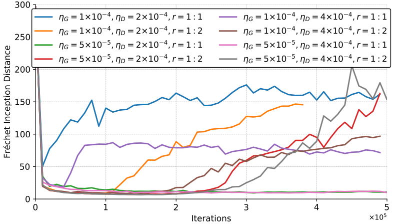
– Always diverges.

– Many hyperparameters ($\eta_G, \eta_D, \beta_1, \gamma, r$).

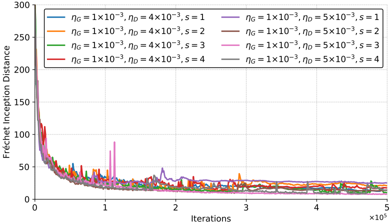
\+ if convergence $\rightarrow$ fast

Experiments
SVRE yields stable GAN optimization

Stochastic baseline



SVRE



– Always diverges.
– Many hyperparameters ($\eta_G, \eta_D, \beta_1, \gamma, r$).
+ if convergence $\rightarrow$ fast

+ Does not diverge.
+ fewer hyperparameters (omits β_1, γ, r)
– slower for very deep nets.

SVRE: Takeaways

SVRE: Takeaways

- Controlling variance is more critical for games (could be reason behind success of *Adam* on GANs)
- SVRE: combines Extragradient and variance reduction.
- Best convergence rate (under some assumptions) for games.
- Good stability properties.

THANKS.

Questions?

References I

A. Brock, J. Donahue, and K. Simonyan. Large Scale GAN Training for High Fidelity Natural Image Synthesis. ArXiv e-prints, September 2018.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in Neural Information Processing Systems 27, pages 2672–2680. 2014.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In NIPS, 2016.

Appendix

The GAN framework

Equilibrium at $p_g = p_d$

The discriminator maximizes:

$$
\begin{aligned}
V(G, D) &= \int_x p_d(x) \log(D(x)) \, \mathrm{d}x + \int_z p_z(z) \log(1 - D(G(z))) \, \mathrm{d}z \\
&= \int_x p_d(x) \log(D(x)) + p_g(x) \log(1 - D(x)) \, \mathrm{d}x
\end{aligned}
$$

Where we used $x = G(z)$, and p_g is the distribution of x.
Hence, the optimal discriminator D^* is:

$$\boxed{D^*(x) = \frac{p_d(x)}{p_d(x) + p_g(x)}}$$

The GAN framework

Equilibrium at $p_g = p_d$

The generator minimizes:

$$
\begin{aligned}
V(G, D^*) &= \mathop{\mathbb{E}}_{x\sim p_d}[\log D^*(x)] + \mathop{\mathbb{E}}_{x\sim p_g}[\log(1 - D^*(x))] \\
&= \mathop{\mathbb{E}}_{x\sim p_d}[\log \frac{p_d(x)}{p_d(x) + p_g(x)}] + \mathop{\mathbb{E}}_{x\sim p_g}[\log \frac{p_g(x)}{p_d(x) + p_g(x)}] \\
&= -\log 4 + \mathbb{D}_{KL}(p_d || \frac{p_d + p_g}{2}) + \mathbb{D}_{KL}(p_g || \frac{p_d + p_g}{2}) \\
&= -\log 4 + 2 \cdot \mathbb{D}_{JS}(p_d || p_g)
\end{aligned}
$$

where we used: $\mathbb{D}_{JS}(p\|q) = \frac{1}{2}\mathbb{D}_{KL}(p\|\frac{p+q}{2}) + \frac{1}{2}\mathbb{D}_{KL}(q\|\frac{p+q}{2})$.

The optimum is reached when $p_g = p_d$,
and the optimal value is $-\log 4$.